EN2550 - Assignment 2

Name – H.M.P.Siriwardana

Index – 190595J

GitHub link -
https://github.com/Maleeshas/Assignment_Image_Processing_EN2550/blob/main/Assignment_02_190595J.ipynb

1)

The circle by RANSAC and the RANSAC function is given below,

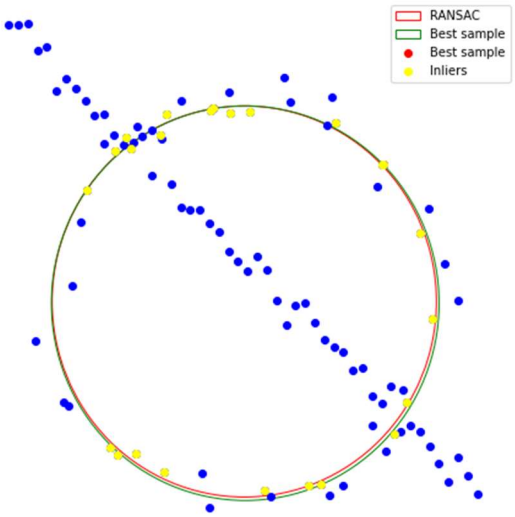
```
def Circle_RANSAC(points):
    threshold = np.std(points)/16
    num_iter = np.log(1 - 0.95)/np.log(1 - (1 - 0.5)**3)
    iter_completed, max_inlierc, selected_model = 0, 0, None

    while iter_completed < num_iter:
        iter_completed += 1
        np.random.shuffle(points)
        sample_points = points[:3]
        xc,yc,radius,_ = cf.least_squares_circle((sample_points))
        center = (xc, yc)
        error = np.abs(radius - np.sqrt(np.sum((center - points[3:])**2, axis=1)))
        inliers = error <= threshold
        inlier_c = np.count_nonzero(inliers)


        if inlier_c > max_inlierc:
            max_inlierc = inlier_c
            inlier_points = []

            for index, inlier in enumerate(inliers):
                if inlier == True:
                    inlier_points.append(points[3:][index])
            inlier_points = np.array(inlier_points)
            selected_model = (center, radius, sample_points, inlier_points)


    xc,yc,radius,_ = cf.least_squares_circle(np.concatenate((selected_model[2], selected_model[3]), axis=0))
    best_m = ((xc, yc), radius, selected_model[2], selected_model[3])
    return best_m
```

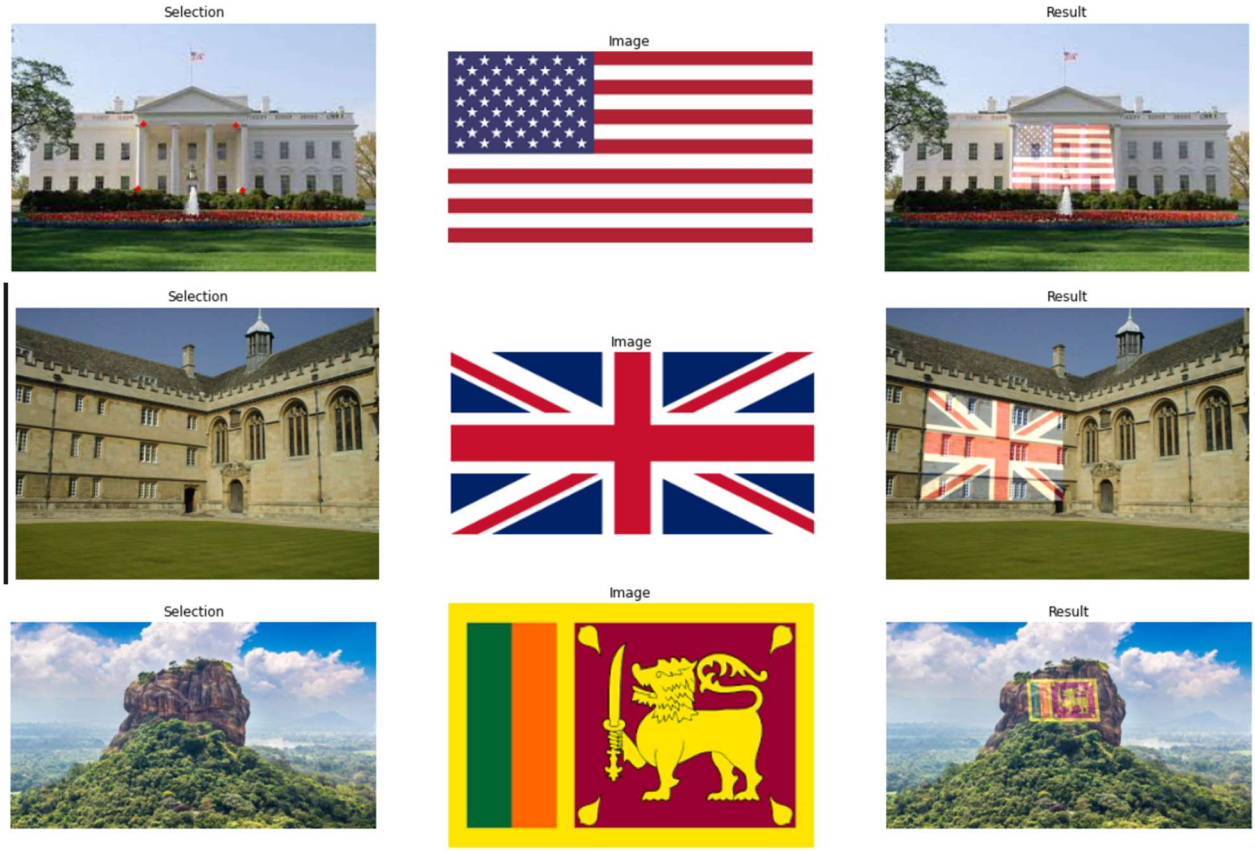


2)The code for this is given in the GitHub code,

Whitehouse with American Flag

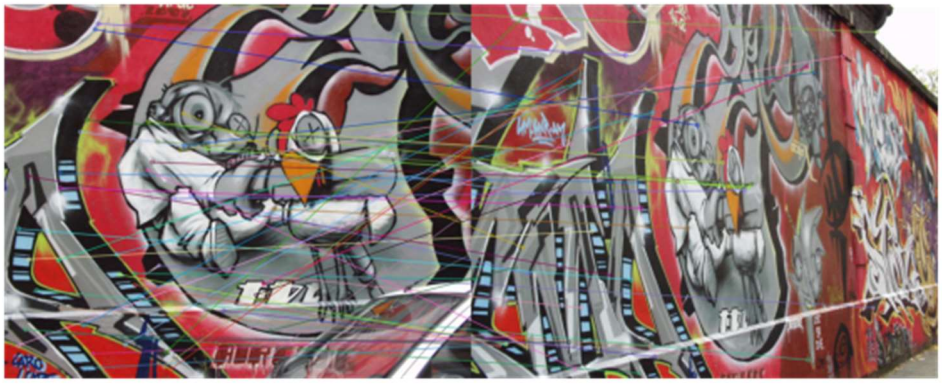
Wadham College with British Flag

Sigiriya with Sri Lankan Flag



3)

a) Feature mapping



b) Main functions of the code,

Homography H1to5 is calculated by,

- Calculating H1to2,H2to3,H3to4,H4to5 homographies.
- Multiplying H1to2,H2to3,H3to4,H4to5 to get H1to5.

```
def homography(X, Y):
    O = np.array([[0],[0],[0]])

    A = []

    for i in range(4):
        A.append(np.concatenate((O.T, np.expand_dims(X.T[i,:], axis=0), np.expand_dims(-1*Y[1, i]*X.T[i,:], axis=0) ), axis=1))
        A.append(np.concatenate((np.expand_dims(X.T[i,:], axis=0), O.T, np.expand_dims(-1*Y[0, i]*X.T[i,:], axis=0) ), axis=1))

    A = np.array(A).squeeze().astype(np.float64)

    e_vals, e_vectors = np.linalg.eig(A.T @ A)
    H = e_vectors[:, np.argmin(e_vals)]
    H = H.reshape(3, -1)

    return H
def inlier_c(X_full, Y_full, H, t, X_inliers, Y_inliers):
    count = 0
    t_X_full = H @ X_full
    t_X_full = t_X_full / t_X_full[2,:]

    error = np.sqrt(np.sum(np.square(t_X_full - Y_full), axis=0))
    inlier_indices = np.where(error <= t)[0]

    X_inliers = np.concatenate((X_inliers, X_full[:,inlier_indices]), axis=1)
    Y_inliers = np.concatenate((Y_inliers, Y_full[:,inlier_indices]), axis=1)

    count = inlier_indices.shape[0]
    return count, X_inliers, Y_inliers
```

```python
def sift_f(img1p, img2p):
    img1 = cv.imread(img1p)
    img5 = cv.imread(img2p)

    img1_gray = cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
    img5_gray = cv.cvtColor(img5, cv.COLOR_BGR2GRAY)

    sift = cv.SIFT_create(nOctaveLayers = 3,contrastThreshold = .1,edgeThreshold = 25,sigma =1)

    kp1, dp1 = sift.detectAndCompute(img1_gray, None)
    kp2, dp2 = sift.detectAndCompute(img5_gray, None)

    kp_img_1 = np.zeros(img1_gray.shape)
    kp_img_5 = np.zeros(img5_gray.shape)

    kp_img_1 = cv.drawKeypoints(img1_gray, kp1, kp_img_1)
    kp_img_5 = cv.drawKeypoints(img5_gray, kp2, kp_img_5)

    bf = cv.BFMatcher()
    matches = bf.knnMatch(dp1,dp2,k=2)

    gm = []
    for m,n in matches:
        if m.distance < 0.75*n.distance:
            gm.append([m])

    matches_img = np.zeros(img1_gray.shape)
    matches_img_fifty = np.zeros(img1_gray.shape)

    matches_img = cv.drawMatchesKnn(img1_gray, kp1, img5_gray, kp2, gm, None, flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    gm = np.squeeze(np.array(gm))

    return gm, kp1, kp2
```

Comparison of homography ,

```
Homography Calculated:
[[ 6.15148651e-01  5.67280933e-02  2.23121595e+02]
 [ 2.15749882e-01  1.15765278e+00 -2.31113011e+01]
 [ 4.74607274e-04 -3.77424514e-05  1.00000000e+00]]
Homography :
[[ 6.2544644e-01  5.7759174e-02  2.2201217e+02]
 [ 2.2240536e-01  1.1652147e+00 -2.5605611e+01]
 [ 4.9212545e-04 -3.6542424e-05  1.0000000e+00]]
Sum of squared differences:7.452615107508754
```

c) Image Stitching